CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CURSO DE ENGENHARIA DE COMPUTAÇÃO
DEPARTAMENTO DE COMPUTAÇÃO

ALGORITMO E ESTRUTURA DE DADOS II

3ª Avaliação - Trabalho Prático: Soluções para o Problema do Caixeiro Viajante.

Nomes:
Ana Clara Cunha Lopes
Gustavo de Assis
Arthur Bracarense
Professor: Thiago de Souza Rodrigues

Belo Horizonte
2023

1. Introdução:

O problema do Caixeiro Viajante consiste em, dado um conjunto de cidades onde existe um caminho entre cada par de cidade com uma distância positiva, encontrar um caminho que, a partir de uma cidade, visita-se todas as cidades e retorna à cidade inicial percorrendo a menor distância possível.

2. Desenvolvimento:

O problema do caixeiro é um exemplo de problema de otimização combinatória, e é um problema considerado NP-Difícil. Esse problema pode ser resolvido por um método força bruta ou um método heurístico.

2.1. Método Força Bruta:

O método força bruta consiste em um algoritmo que determina todas as possíveis rotas e escolhe a de menor distância. Esse método se trata de um método exato, ou seja, o resultado obtido é certamente o melhor resultado, no entanto, esse método tem complexidade fatorial, uma vez que o número de rotas entre n cidades é (n-1)!.

Por exemplo: Considerando 4 cidades: A,B,C e D. Sendo a primeira a A, temos 3 outras cidades para ir: B, C e D, escolhendo uma delas, temos mais duas cidades para ir, e depois, só mais uma para ir (tendo em vista que da última só podemos voltar à primeira). Logo, temos que para 4 cidades, temos 3*2*1 = 3! = 6 rotas.

Para calcular todos os caminhos possíveis e determinar o caminho com o menor custo, você pode usar um algoritmo de busca em profundidade (DFS) para explorar todos os caminhos possíveis e calcular o custo de cada caminho no processo. A implementação usada foi uma implementação modificada do algoritmo do Ziviane.

O programa se baseia em criar instâncias de 2 até 12 cidades e aplicar o algoritmo de força bruta para achar o melhor caminho. O tempo é contabilizado, porém, existem muitos fatores que podem alterar o tempo de execução de determinado algoritmo (operações do SO em segundo plano, dentre outros). Para amenizar esse problema, foram feitos 10 testes e calculado a média entre eles. Os dados obtidos são mostrados a seguir:

```
O tempo para n = 2 cidades é de: 0.0 segundos
O tempo para n = 3 cidades é de: 0.0 segundos
O tempo para n = 4 cidades é de: 0.0 segundos
O tempo para n = 5 cidades é de: 0.0 segundos
O tempo para n = 6 cidades é de: 0.0 segundos
O tempo para n = 7 cidades é de: 0.0 segundos
O tempo para n = 8 cidades é de: 0.0 segundos
O tempo para n = 9 cidades é de: 0.006 segundos
O tempo para n = 10 cidades é de: 0.049 segundos
O tempo para n = 11 cidades é de: 0.468 segundos
O tempo para n = 12 cidades é de: 5.444 segundos
```

Considerando os tempos muito abaixo de 0 como irrelevantes, temos que os dados podem ser representados pela tabela:

n	Tempo
2	0,0
3	0,0
4	0,0
5	0,0
6	0,0
7	0,0
8	0,0
9	0,006
10	0,049
11	0,468
12	5,444

Com isso podemos fazer nosso gráfico, obtendo:

Junto a isso, procurando em bibliografias auxiliares temos que:

n	rotas por segundo	(n - 1)!	cálculo total
5	250 milhoes	24	insignific
10	110 milhoes	362 880	0.003 seg
15	71 milhoes	87 bilhoes	20 min
20	53 milhoes	1.2×10^{17}	73 anos
25	42 milhoes	6.2×10^{23}	470 milhoes de anos

Ambos dados ressaltam o crescimento exponencial do problema do caixeiro viajante.

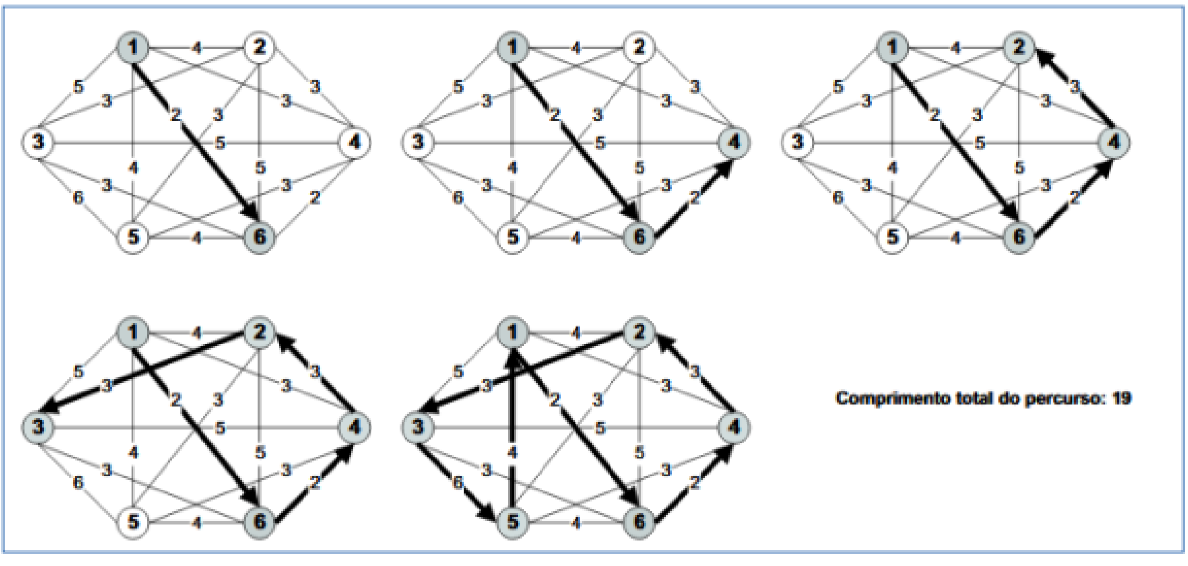
Podemos então determinar que o método força bruta é aplicável somente se o problema envolver poucas cidades. Caso o problema envolva mais cidades, é necessário aplicar um método heurístico.

2.2. Método Heurístico:

Métodos Heurísticos são métodos não exatos, mas que geram respostas satisfatórias em casos em que métodos exatos são inviáveis. Esses métodos se baseiam em uma intuição (regras) para resolver o problema. No caso do problema do caixeiro viajante, os métodos heurísticos descartam rotas não promissoras, seguindo um caminho considerado ideal e por fim achando uma boa solução.

Existem muitos métodos heurísticos para solução do problema do caixeiro viajante, sendo alguns deles: a inserção do vizinho mais próximo, a inserção mais barata, algoritmo de Christofides, etc. Nesse trabalho usaremos a heurística da inserção do vizinho mais próximo, não necessariamente o melhor método, mas um dos mais simples para ser aplicado.

Esse método consiste em construir uma rota adicionando, a cada passo, o nó mais próximo do último nó inserido e que ainda não tenha sido visitado.



Exemplo: https://docs.ufpr.br/~volmir/PO_II_12_TSP.pdf

Considerando que sempre partimos da primeira cidade (vértice número 0), o problema do caixeiro viajante para as instâncias fornecidas são:

- **si535:**

Esse é um problema que possui 535 cidades, sendo que as distâncias estão disponíveis em forma de matriz de adjacência, mas somente a diagonal superior desta matriz.

Caminho:

0 8 20 135 486 502 509 521 529 510 497 481 478 491 511 518 531 201 226 240 246 257 234 228 212 247 213 227 238 244 263 237 224 210 242 259 527 515 514 490 485 488 499 513 516 532 524 534 279 319 146 341 288 317 337 287 310 338 290 321 334 291 309 340 137 139 141 143 136 138 140 142 144 145 171 172 173 174 148 149 327 316 280 273 271 278 320 328 272 281 318 332 277 270 308 150 151 333 284 307 329 254 248 232 216 207 206 217 236 251 258 218 204 25 533 26 520 27 507 496 483 23 22 21 492 476 468 10 28 428 422 409 437 32 447 11 454 469 457 445 429 411 414 435 443 12 36 453 471 470 465 433 415 24 9 475 503 31 508 30 523 29 528 209 215 241 243 262 233 221 211 245 264 33 526 34 519 35 505 501 489 487 494 39 506 38 517 37 530 13 40 371 375 373 372 364 2 362 368 376 363 374 410 430 417 436 412 440 449 451 466 459 464 458 448 439 421 418 431 461 463 480 498 479 495 477 493 202 222 239 252 261 230 220 214 203 219 229 250 260 249 256 205 225 235 253 255 231 223 208 268 282 313 331 285 269 312 330 289 314 335 286 311 147 339 283 315 336 525 522 512 504 484 482 500 419 427 441 450 460 456 444 438 416 420 432 446 455 472 452 442 434 413 462 467 473 365 366 370 367 369 1 3 4 180 179 5 360 378 380 377 381 379 388 389 395 6 382 383 387 386 384 401 403 402 404 391 390 392 393 394 405 406 407 408 361 396 400 397 398 399 358 357 359 425 426 124 121 120 123 122 192 119 196 197 125 195 128 198 199 188 131 189 126 191 190 127 130 133 129 132 423 424 114 110 106 102 101 105 100 104 108 111 193 107 103 99 161 163 165 115 194 112 116 118 113 109 117 95 91 96 93 88 85 89 94 98 97 86 81 80 84 90 83 79 87 92 157 153 82 74 69 64 59 54 55 60 65 70 75 66 61 56 57 62 67 63 58 68 73 78 72 77 71 76 18 51 50 52 53 17 46 47 48 16 44 45 42 43 49 185 184 182 181 183 186 187 14 41 15 296 297 299 300 301 322 323 344 343 342 324 325 326 345 152 304 303 306 305 302 349 350 348 347 346 295 294 293 292 298 353 354 352 351 355 356 275 170 267 265 266 274 276 134 200 474 19 385 7 155 158 159 154 156 160 162 164 166 167 168 169 178 177 176 175 0

Distância Total do Caminho: 50144

- **pa561:**

Esse é um problema que possui 561 cidades, sendo que as distâncias estão disponíveis em forma de matriz de adjacência, mas somente a diagonal inferior desta matriz.

Caminho:

0 323 330 324 329 331 328 326 325 327 399 157 158 480 481 159 160 163 161 155 162 112 111 116 119 115 118 122 121 125 184 185 127 126 128 129 130 131 123 124 120 164 138 165 168 170 141 144 133 139 140 135 134 132 186 189 187 188 94 96 97 95 93 92 72 90 91 69 52 53 54 49 46 42 41 171 172 173 47 174 175 113 109 106 107 108 153 105 152 154 156 479 167 169 143 145 146 147 227 228 229 230 506 508 521 520 509 519 513 512 511 495 497 498 514 515 518 517 458 459 460 445 444 437 436 416 413 410 409 492 486 485 408 405 407 411 406 423 422 403 402 397 417 334 419 336 335 350 337 352 355 338 340 339 341 420 421 425 426 428 429 432 434 431 435 412 430 424 418 398 401 400 404 482 483 484 487 488 489 490 491 494 496 493 414 499 516 415 462 461 466 446 451 447 387 448 391 390 369 389 388 385 386 442 384 440 441 438 439 382 380 383 363 364 365 368 367 366 370 371 373 372 374 375 376 377 378 396 395 393 394 449 450 453 454 455 456 457 478 468 452 476 477 475 472 471 470 469 467 465 464 463 543 542 541 524 523 522 231 507 529 531 532 251 268 269 270 267 248 249 250 232 233 234 236 237 235 238 247 265 245 244 243 242 241 240 255 256 253 258 257 259 260 317 316 315 314 313 318 312 320 302 303 301 305 306 308 304 321 319 322 311 276 275 273 272 271 287 288 279 291 290 289 294 293 298 292 295 296 297 299 300 307 310 309 264 261 262 263 274 266 239 246 222 221 151 148 142 136 150 137 215 216 214 191 192 193 195 194 196 209 208 207 206 205 204 102 98 99 79 80 100 81 82 87 86 85 84 83 73 64 61 60 59 58 57 67 66 75 77 89 68 51 44 31 40 36 35 4 5 7 8 6 18 15 16 17 19 11 12 28 29 21 33 32 34 22 30 23 24 25 26 13 14 20 27 10 9 2 1 38 37 3 39 43 45 50 48 55 70 71 56 180 181 182 183 179 177 176 117 178 114 110 190 218 219 225 197 200 201 203 199 210 211 212 103 88 104 101 76 78 74 65 62 63 342 343 344 345 348 347 346 332 333 349 351 353 357 354 358 356 361 359 360 362 379 427 433 381 443 392 549 548 473 474 550 551 552 553 554 556 555 547 546 525 526 538 539 540 557 558 559 560 536 537 530 527 528 534 533 535 281 286 285 284 283 282 280 277 278 254 223 224 220 217 202 198 252 226 213 149 500 501 502 503 504 505 510 544 545 166 0

Distância Total do Caminho: 3422

- **si1032:**

Esse é um problema que possui 1032 cidades, sendo que as distâncias estão disponíveis em forma de matriz de adjacência, mas somente a diagonal superior desta matriz.

Caminho:

0 1 2 72 73 74 144 145 146 147 240 241 242 243 336 337 338 423 451 461 480 526 529 488 456 454 425 341 340 339 247 246 245 244 151 150 149 148 77 76 75 5 4 3 6 7 8 78 79 80 152 153 154 155 248 249 250 251 342 343 344 427 452 460 501 520 535 492 464 449 430 347 346 345 255 254 253 252 159 158 157 156 83 82 81 11 10 9 12 13 14 84 85 86 160 161 162 163 256 257 258 259 348 349 350 420 447 467 493 522 549 495 479 453 418 353 352 351 263 262 261 260 167 166 165 164 89 88 87 17 16 15 18 19 20 90 91 92 168 169 170 171 264 265 266 267 354 355 356 415 448 476 502 521 543 489 477 450 413 359 358 357 271 270 269 268 175 174 173 172 95 94 93 23 22 21 24 25 26 96 97 98 176 177 178 179 272 273 274 275 360 361 362 410 446 475 483 518 532 485 466 455 428 365 364 363 279 278 277 276 183 182 181 180 101 100 99 29 28 27 30 31 32 102 103 104 184 185 186 187 280 281 282 283 366 367 368 419 443 478 482 504 545 487 474 442 409 371 370 369 287 286 285 284 191 190 189 188 107 106 105 35 34 33 506 544 519 542 525 541 523 546 527 540 524 533 750 740 745 739 743 748 741 742 744 746 747 749 751 731 738 722 753 772 775 781 780 791 782 785 771 755 752 759 765 767 757 827 822 819 824 829 831 825 821 820 823 826 828 818 816 817 793 794 795 796 797 798 799 800 801 802 803 792 805 806 807 808 809 810 811 812 813 814 815 804 760 763 761 766 768 762 769 758 756 754 764 783 788 786 787 789 790 784 770 779 778 777 776 774 773 721 720 852 855 858 850 863 866 869 840 832 836 839 842 845 847 830 901 904 909 903 899 907 911 902 905 900 910 897 898 882 877 881 880 879 878 872 876 875 874 873 883 885 886 887 888 889 890 891 892 893 894 884 895 896 844 843 841 837 838 849 835 833 834 848 851 871 868 867 865 864 862 870 860 859 861 856 854 853 857 927 932 933 935 936 937 938 940 930 951 950 942 947 945 931 928 916 914 915 929 919 918 921 922 924 925 975 964 974 973 972 971 970 969 968 967 966 965 952 962 961 960 959 958 957 956 955 954 963 953 977 976 991 990 979 988 987 986 978 985 984 983 982 981 980 923 920 912 913 917 943 946 949 944 939 941 934 948 995 583 1027 584 1006 585 999 586 1001 587 1023 588 1015 589 1022 573 1014 569 1016 571 1018 570 1020 553 1021 554 1024 555 1026 556 1028 557 1029 558 1011 559 1010 560 994 552 996 562 998 563 992 564 1002 565 1003 566 1005 567 1007 561 1009 568 615 604 613 612 611 610 609 618 608 607 606 605 602 603 593 594 595 596 592 598 599 600 597 601 617 614 629 630 619 628 624 622 620 626 621 623 625 627 631 574 1004 591 1008 1000 575 997 576 993 577 1030 582 1031 578 1025 579 1012 580 1017 581 1013 572 1019 590 635 651 652 654 655 657 658 660 650 671 670 668 662 665 667 653 648 634 636 637 640 642 643 646 641 644 632 616 687 686 681 685 684 683 682 673 680 679 678 677 676 675 674 672 691 690 694 711 709 708 707 706 705 704 703 710 693 701 700 692 699 698 702 697 696 695 638 647 645 639 649 633 663 666 669 664 659 656 661 718 719 712 717 716 715 714 713 688 689 989 846 725 734 733 732 730 729 728 727 726 723 724 735 736 737 906 908 926 528 511 490 459 438 426 404 403 402 331 330 329 328 235 234 233 232 140 139 138 68 67 66 63 64 65 135 136 137 228 229 230 231 324 325 326 327 399 400 401 422 434 457 491 516 530 481 469 445 429 398 397 396 323 322 321 320 227 226 225 224 134 133 132 62 61 60 57 58 59 129 130 131 220 221 222 223 316 317 318 319 393 394 395 431 436 462 503 514 534 500 470 435 408 392 391 390 315 314 313 312 219 218 217 216 128 127 126 56 55 54 51 52 53 123 124 125 212 213 214 215 308 309 310 311 387 388 389 421 439 465 497 512 531 496 468 433 417 386 385 384 307 306 305 304 211 210 209 208 122 121 120 50 49 48 45 46 47 117 118 119 204 205 206 207 300 301 302 303 381 382 383 416 432 458 486 510 550 498 471 444 414 380 379 378 299 298 297 296 203 202 201 200 116 115 114 44 43 42 39 40 41 111 112 113 196 197 198 199 292 293 294 295 375 376 377 412 437 473 499 508 547 494 472 440 411 374 373 372 291 290 289 288 195 194 193 192 110 109 108 38 37 36 505 551 509 548 507 539 513 536 515 538 406 405 335 334 333 332 239 238 237 236 143 142 141 71 70 69 407 424 441 463 484 517 537 0

Distância Total do Caminho: 94571

3. **Conclusão:**

Temos que apesar dos métodos heurísticos serem não exatos, eles se mostram uma possibilidade para problemas em que os métodos exatos não são capazes de dar a resposta em tempo hábil. No caso, conseguimos aplicar métodos exatos para resolução do problema do caixeiro viajante até cerca de 15 cidades, no entanto, acima disso o tempo necessário para os cálculos se torna grande demais. Por sua vez, com métodos heurísticos podemos calcular uma solução para esse problema considerando centenas de cidades, e ainda termos uma resposta satisfatória.

4. Referências Bibliográficas:

PORTO, Silvio. O problema do caixeiro viajante. Disponível em: <http://www.mat.ufrgs.br/~portosil/caixeiro.html>. Acesso em: [10/12/2023].

POZSAR, Volmir. Problema do Caixeiro Viajante - Programação Orientada a Objetos II. Disponível em: <https://docs.ufpr.br/~volmir/PO_II_12_TSP.pdf>. Acesso em: [10/12/2023].

Marcos. Ciência da Computação - Estruturas de Dados. Curitiba: UFPR, 2023. Disponível em: <https://www.inf.ufpr.br/marcos/ci242/aula4.pdf>. Acesso em: [10/12/2023].

ZIVIANI, Nivio. Projeto de Algoritmos com Implementação em Java. 2. ed. São Paulo: Cengage Learning, 2007.